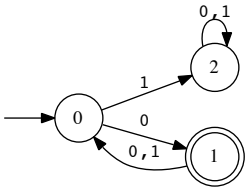
Université d'Orléans — **Master IRAD**
Logiques et leurs applications — *Corrigé du devoir Surveillé* — **Année 08-09**
Yohan Boichut

Exercice 1.

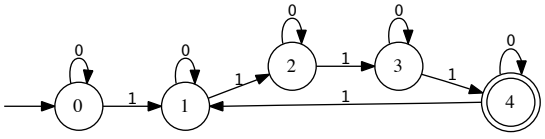
1. Pour calculer l'automate de $\neg\phi$, il suffit de calculer le complément de $\mathcal{A}_\phi$.
2. La cylindrification a pour but dans le cadre de la construction d'un automate pour une formule donnée d'homogénéiser l'ensemble des variables d'un point de vue automate pour les conjonctions ou disjonctions de sous-formules directes. Par exemple, soit $\phi(X_1,\ldots,X_n,Y_1,\ldots,Y_m) = \phi_1(X_1,\ldots,X_n)\vee\phi_2(Y_1,\ldots,Y_m)$ et soient $\mathcal{A}_{\phi_1(X_1,\ldots,X_n)}$ et $\mathcal{A}_{\phi_2(Y_1,\ldots,Y_m)}$ les automates reconnaissant respectivement les solutions des sous-formules $\phi_1(X_1,\ldots,X_n)$ et $\phi_2(Y_1,\ldots,Y_m)$. Nous devons cylindrifier $\mathcal{A}_{\phi_1(X_1,\ldots,X_n)}$ afin qu'il puisse reconnaître les solutions de $\phi_1(X_1,\ldots,X_n,Y_1,\ldots,Y_m)$. Symétriquement pour $\mathcal{A}_{\phi_2(Y_1,\ldots,Y_m)}$ avec les variables $X_1,\ldots,X_n$. Les automates obtenus seront $\mathcal{A}_{\phi_1(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}$ et $\mathcal{A}_{\phi_2(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}$. Ainsi, l'automate reconnaissant les solutions de la formule $\phi(X_1,\ldots,X_n,Y_1,\ldots,Y_m)$ est obtenu en faisant l'union de $\mathcal{A}_{\phi_1(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}$ et $\mathcal{A}_{\phi_2(X_1,\ldots,X_n,Y_1,\ldots,Y_m)}$. Pour $\phi(X_1,\ldots,X_n,Y_1,\ldots,Y_m) = \phi_1(X_1,\ldots,X_n)\wedge\phi_2(Y_1,\ldots,Y_m)$, l'opération de cylindrification est identique mais l'opération finale est l'intersection et non l'union.
3. La projection est utilisée lorsque la formule est de la forme $\phi(X_1,\ldots,X_n) = \exists Y.\phi_1(X_1,\ldots,X_n,Y)$. Pour l'automate reconnaissant les solutions de $\phi_1(X_1,\ldots,X_n,Y)$ noté ici $\mathcal{A}_{\phi_1(X_1,\ldots,X_n,Y)}$, il suffit de calculer le $n+1$ projeté de cet automate (supprimer la variable Y) pour obtenir $\mathcal{A}_{\phi(X_1,\ldots,X_n)}$.

Exercice 2. Pour enlever toute ambiguïté, les étiquettes sont localisées au dessus des transitions pour tous les automates. `ALL` est utilisé pour exprimer tous les triplets possibles pour un soucis de lisibilité.

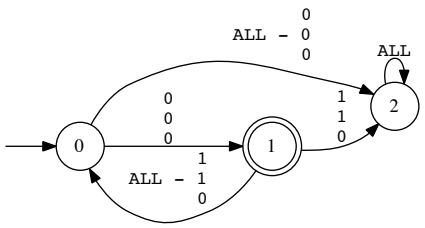
1. Automate reconnaissant si un ensemble contient uniquement des entiers impairs



2. Automate reconnaissant si un ensemble est non vide et a une taille multiple de 4.



3. Automate reconnaissant les triplets $\langle X,Y,Z\rangle$ où X,Y,Z des ensembles d'entiers impairs tels que $X\cap Y\subseteq Z$. X,Y,Z sont ordonnés dans cet ordre mais *verticalement* dans l'automate.



Exercice 3.

1. Entier multiple de 3 :

```
pred formule1(var1 x) =
   ex2 Q: x in Q
     & (all1 q:
        (0 < q & q <= x) =>
            (q in Q <=> ((q - 1 notin Q) & (q-2 notin Q))))
     & 0 in Q;
```

2. X ensemble non vide contenant que des multiples de 3

```
pred formule2(var2 X) =
   X~= empty & (all1 x: x in X => formule1(x)) ;
```

3. X contient uniquement 3 éléments

```
pred formule3(var2 X) =
    ex1 x,y,z: x~=y & x~=z & y~=z & x in X & y in X & z in X &
    (all1 u: u in X => (u=x | u=y | u=z));
```

Exercice 4.

1. a^*bc^* :

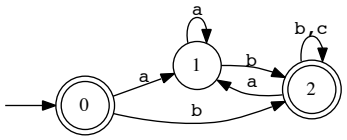
```
pred question4(var2 Xa, Xb, Xc) =

  (ex1 b, fin: (b in Xb & b < fin & (all1 a : ((a < b) => (a in Xa) )) &
        (all1 c : (((b < c) & (c < fin))=> (c in Xc) )) &
(all1 x : ((fin <= x) => (x notin (Xa union Xb union Xc)))))) &
      Xa inter Xb =empty & Xa inter Xc =empty & Xc inter Xb =empty;
```

2. $abb(a+c)^*bba$

```
pred question5(var2 Xa, Xb, Xc) =
   0 in Xa & 1 in Xb & 2 in Xb &
   (ex1 fin: fin > 2 &  fin in Xb & (fin+1 in Xb) &  (fin+2 in Xa) &
      (all1 x: (x > (fin+2)) => (x notin (Xa union Xb union Xc))) &
      (all1 coua: (((coua > 2) & (coua < fin)) => (coua in (Xa union Xc)))))
   &  Xa inter Xb =empty & Xa inter Xc =empty & Xc inter Xb =empty;
```

3. Et enfin, $(a^*bc^*)^*$. Pour cet exemple, nous allons utiliser l'approche entrevue durant le dernier cours : comment construire une formule à partir d'un automate ? Il existe au moins une solution que vous pouviez trouver, mais avec cette approche, à partir du moment où vous parvenez à construire l'automate pour une expression régulière donnée, vous pouvez spécifier n'importe quelle expression

régulière par une formule. (mais pourquoi nous a t'il pas dit çà avant ? Ben, en fait, c'était écrit noir sur blanc dans la preuve du théorème de Büchi présente sur les slides que vous avez lus et relus :-)). Tout d'abord, voici un automate dont le langage respecte l'expression régulière donnée ici.



Dans la formule, nous ferons référence aux états 0, 1 et 2 par Q0, Q1 et Q2.

```
pred pasDeTrou(var2 S) =
all1 x: ((x in S => (all1 y: (y<x) => (y in S))))
;


pred elementMaximum(var1 m, var2 S)=
m in S & (all1 x:( (x>m) => x notin S));

pred question6(var2 Xa,Xb,Xc)=
(ex2 Q0, Q1, Q2: (((Q0 inter Q1) = empty) & ((Q0 inter Q2) = empty) &
((Q1 inter Q2) = empty) & pasDeTrou(Q0 union Q1 union Q2) &
((Xa inter Xb) =empty) & ((Xa inter Xc) =empty) & ((Xc inter Xb) =empty) &
pasDeTrou(Xa union Xb union Xc) &
# etat initial
0 in Q0 &
# transitions partant de Q0
(all1 x: (x in Q0) => (
     (x in Xa => ((x+1) in  Q1)) &
     (x in Xb => ((x+1) in  Q2)) &
     (x notin Xc)))  &
# transitions partant de Q1
(all1 x: (x in Q1) =>
    (((x in Xa) => ((x+1) in  Q1)) &
     ((x in Xb) => ((x+1) in  Q2)) &
       x notin Xc)) &
# transitions partant de Q2
(all1 x: (x in Q2) => (
     (( x in Xa) => ((x+1) in  Q1)) &
     (( x in Xb) => ((x+1) in  Q2)) &
     (( x in Xc) => ((x+1) in  Q2)))) &
#acceptation d'un mot
((ex1 m: (elementMaximum(m,(Xa union Xb union Xc)) & (m+1) in (Q2))) |
  ((Xa = empty) & (Xb = empty) & (Xc = empty)))))
;
```